Spot the Odd Man Out: Exploring the Associative Power of Lexical Resources

Gabriel Stanovsky[1,2] and Mark Hopkins[3]

[1]Allen Institute for Artificial Intelligence, Seattle, WA
[2]Paul G. Allen School of Computer Science & Engineering, University of Washington, Seattle, WA
[3]Department of Mathematics, Reed College, Portland, OR
gabis@allenai.org, hopkinsm@reed.edu

Abstract

We propose Odd-Man-Out, a novel task which aims to test different properties of word representations. An Odd-Man-Out puzzle is composed of 5 (or more) words, and requires the system to choose the one which does not belong with the others. We show that this simple setup is capable of teasing out various properties of different popular lexical resources (like WordNet and pre-trained word embeddings), while being intuitive enough to annotate on a large scale. In addition, we propose a novel technique for training multi-prototype word representations, based on unsupervised clustering of ELMo embeddings, and show that it surpasses all other representations on all Odd-Man-Out collections.

1 Introduction

Correctly disambiguating the sense of a polysemous word (e.g., "***spring*** *is a beautiful season*" versus "*John was ready to* ***spring*** *into action*") is a crucial part of various NLP tasks, such as translation, question answering, or textual entailment. The state-of-the-art, and the de-facto common practice for essentially all of these tasks, involves neural networks (see (Goldberg, 2015) for a recent survey), which are commonly initialized with pre-trained word vectors, such as Word2Vec (Mikolov et al., 2013a) or GloVe (Pennington et al., 2014).

These widely-used representations often significantly improve performance in downstream tasks, as they are able to leverage large amounts of unstructured data. However, most of the popular collections of word embeddings assign only one vector to each word, thus shifting the burden of word disambiguation to deeper, task-specific layers, which commonly rely on data of much smaller scales.

While there has been a significant body of work around *sense embeddings* (i.e., embedding senses, instead of lexical units), evaluating such representations remains the subject of debate (Faruqui et al., 2016; Gladkova and Drozd, 2016).

In this work, we propose a new evaluation task called Odd-Man-Out. The goal of an Odd-Man-Out puzzle is simple. Given a set of words[1] like **cherry, orange, apple, grass, grape**, the objective is to identify the word that does not belong (here, the answer is **grass**, because it is not a fruit). While there are often multiple relationships among the words, we will show that non-experts typically agree on the odd-man-out, and are able to generate hard puzzles on a large scale, using a novel crowdsourcing protocol.

Following the creation of this large test set, we conduct a thorough analysis of the ability of various lexical resources to correctly solve the task. In doing so, we embrace the suggestion of Gladkova and Drozd (2016), which argue for "a shift from absolute ratings of word embeddings towards more exploratory evaluations that would aim not for generic scores, but for identification of strengths and weaknesses of embeddings", and conduct rigorous analysis of each representation. Overall, we find that all lexical resources are prone to miss associations which are intuitive for humans, leaving ample room for future improvement. Moreover, we show empirical evidence that lexical resources that do not account for polysemy are handicapped by this weakness.

Finally, we propose a new sense embedding technique, which leverages the recent introduction of ELMo embeddings (Peters et al., 2018) by performing unsupervised clustering over a large unstructured corpus. We show that this new resource surpasses all other baselines on various Odd-Man-Out datasets.

[1]In this paper, we use the term "word" loosely to also include the multi-word expressions like "fire engine" and "magnifying glass."

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1533–1542
Brussels, Belgium, October 31 - November 4, 2018.

We make all our code and models publicly available.[2]

2 Existing Evaluation Methods

In this section, we briefly survey some existing evaluation methods for lexical resources, and discuss their pros and cons.

2.1 Word Similarity

The *word similarity* task (Rubenstein and Goodenough, 1965) has been a dominant approach to assess word vector quality. In this task, systems are required to score the similarity of two words on a numeric scale, sometimes without context and sometimes in a sentential context (Huang et al., 2012). For instance, the pair (tiger, mammal) has a similarity score of 6.85 (out of 10) on the WordSim dataset (Finkelstein et al., 2001).

A common criticism (Faruqui et al., 2016; Gladkova and Drozd, 2016) of the word similarity task is that "similarity" is subjective, and conflates several different potential relationships between words. For instance, (Faruqui et al., 2016) questions why the pair (cup, coffee) should be considered more similar than (car, train), as it is according to the WordSim dataset.

Odd-man-out puzzles naturally disambiguate the nebulous concept of "similarity", because each puzzle implicitly defines a specific relationship. The words "car" and "train" may be similar, but this similarity is irrelevant is the context of a puzzle like **car, train, checkered flag, racetrack, pit stop**.

2.2 Analogies

Analogies like "king is to queen as man is to..." (Mikolov et al., 2013b; Jurgens et al., 2012) are related to the odd-man-out task. Analogies, however, are best suited to particular relationships, such as hypernym → hyponym (which are the subject of extensive research, e.g., (Shwartz et al., 2016)), while odd-man-out puzzles can capture a broader range of associations (see for instance, the auto-racing puzzle from the previous section). Analogies are also more subject to ambiguity, since the premise can involve only two words. For instance, the puzzle "cherry is to strawberry as grass is to..." could refer to the fact that cherry and strawberry are both fruits, both red, or both red fruits. Odd-man-out puzzles provide a simple way of reducing ambiguity: adding more choices to the puzzle.

2.3 Word Sense Disambiguation and Induction

Word sense disambiguation (Navigli, 2009) is a popular way to evaluate polysemous word representations. The common criticism is that systems are rewarded based on their ability to classify words according to a fixed inventory of senses, whose granularity is regarded by some as too coarse and others as too fine. An alternative is word sense induction (Manandhar et al., 2010), which allows systems to cluster word senses without an agreed-upon sense inventory. However, there is not an obvious evaluation metric. The two metrics used in SemEval 2010 Task 14 (Manandhar et al., 2010) yielded highly divergent system rankings.

2.4 Word Context Relevance

A recent evaluation method is Word Context Relevance (Arora et al., 2016; Sun et al., 2017a). The task is to identify whether a particular bag of words is "relevant" to a target word. For instance, "tie" is considered relevant to the bag "winner, score, tied, completion, identical, results, sports," but irrelevant to the bag "domestic, hog, pig, culinary, eaten, cooked, fat". This task has the attractiveness of being a simple binary evaluation, but demands only that a model can identify a broad sense of relatedness, not the ability to pinpoint specific relationships.

2.5 Lexical Substitution

Lexical substitution tasks (McCarthy and Navigli, 2007; Biemann, 2013; Kremer et al., 2014), in which the task is to determine whether one word can replace another word in a particular context, are effective, but are restricted in the kind of relationships they can test (mainly synonymy).

3 Odd Man Out Datasets

In this section, we describe the creation of several Odd-Man-Out datasets. We begin by describing a small-scale, curated annotation, then show how to scale the annotation using crowdsourcing techniques. In the subsequent sections, we use these datasets to explore the properties of a wide array of lexical resources.

[2] https://github.com/gabrielStanovsky/odd-man-out

3.1 Expert annotation

To create our first odd-man-out datasets, we used categories from the card game Anomia.[3] Anomia is a slapjack-style game in which players attempt to name instances of a particular category as quickly as possible. Categories include: "percussion instrument", "Mexican food", and "Michael Jackson song". A team of 4 people, trained in-house, created one odd-man-out puzzle for each category, yielding a total of 404 puzzles. For instance, the puzzle corresponding to "percussion instrument" is **clarinet, drum, xylophone, tambourine, cymbals**, where clarinet is the odd-man-out. The puzzle-writing team attempted to make the odd-man-out similar to the category, e.g. clarinet is a musical instrument but not a percussion instrument.

We divided the puzzles into 2 sets, one for puzzles comprised of common words (like the "percussion instruments" example above), and one for puzzles comprised of proper nouns (like the puzzle derived from the category "Michael Jackson songs"). Each set contained exactly 202 puzzles. We further divided each of these sets into a development set (used for error analysis) of size 100 and a test set of size 102. Henceforth, we will refer to these four datasets as ANOMIACOMMONDEV, ANOMIACOMMONTEST,ANOMIAPROPERDEV, andANOMIAPROPERTEST.

There are two potential pitfalls in creating these puzzles. First, the underlying category may not be detectable by humans. Second, we may inadvertently create an ambiguous puzzle with multiple odd-men-out. For instance, given the category "vegetables", we might create the puzzle **grass, celery, cucumber, carrot, lettuce**, for which the intended odd-man-out is **grass**, but the answer **carrot** is also a reasonable odd-man-out for the category "things that are green."

To ensure the answerability of our puzzles, we administered the ANOMIACOMMONDEV to five college-educated individuals, three of whom were native English speakers. Both groups did well. The native speakers averaged 95.3% accuracy, while the non-native speakers averaged 91.5%.

[3]https://boardgamegeek.com/boardgame/142271/anomia-party-edition

3.2 Crowdsourcing

Following the success of the curated annotation, as described above, we devised a crowdsourcing protocol to achieve annotation at a much larger scale. In this section, we describe this protocol and show that the Odd-Man-Out task is intuitive enough to be collected and annotated with high agreement by non-trained annotators, using a small expert seed annotation. This enables us to efficiently obtain a large set of 500K *hard* "training" samples on a small budget of $400. Finally, we also validate a set of 1000 puzzles, which we use in following sections for testing purposes.

Crowdsourcing protocol Our semi-automatic crowdsourcing protocol starts from a seed set of about 250 polysemous words,[4] and is composed of the following three consecutive stages:

1. **Seed categorization (expert)**: given a polysemous word w, we ask expert annotators to come up with at least two categories c_1, c_2, which describe w. For example, given w = "*bat*", the corresponding categories can be c_1 = "*nocturnal mammal*" and c_2 = "*baseball instrument*". This categorization was performed by the authors of this paper, and was done in about 3 person hours.

2. **Category expansion (crowdsourced)**: Given a seed word and a corresponding category (w, c) turkers are asked to provide five more examples of the category c , which are similar to, but different than, w. For example, given ("*bat*", "*nocturnal mammal*") turkers are expected to provide examples such as "*beaver*", "*badger*", or "*hedgehog*", while for ("*bat*", "*baseball instrument*"), proper answers include "*ball*" or "*cap*". We used the Amazon Mechanical Turk (AMT)[5] platform, paying 15¢ per elicitation.

3. **Puzzle creation (automatic)**: For each polysemous seed word w, belonging to categories c_1, c_2, we automatically create Odd-Man-Out puzzles by concatenating to w each possible combination of three words from c_1 (c_2), while the intended odd-man-out is any word from c_2 (c_1). For example, using the previous

[4]https://en.wikipedia.org/wiki/List_of_true_homonyms, following (Sun et al., 2017b).

[5]https://www.mturk.com

seed word and categories, we get the following puzzle: ("*cap*", "*bat*", "*beaver*", "*badger*", "*hedgehog*"). Creating puzzles by combining words from c_1 and c_2 in this manner ensures that task is challenging, as it requires to disambiguate the polysemous word (e.g., "*bat*").

Performing the process described above with the 249 polysemous seed words yielded a large corpus of $497,365$ puzzles, thanks to the combination process, with a vocabulary of $1,312$ different words. To create a gold high quality test corpus, and to assess the validity of this semi-automatic process, we sampled 1000 puzzles and administered each puzzle to three annotators on AMT, paying 6¢ per response. We found that for 84.3% of the instances there was a majority vote agreement on the intended odd-man-out word. We refer to this set of 843 puzzles as CROWDSOURCED843. Table 1 shows examples of the crowdsourced puzzles.

From an examination of turkers disagreement, we can attribute the vast majority to noise in one of the annotation stages, e.g., turkers which provide examples of the wrong sense of the seed word, resulting in invalid puzzles (where all of the words belong to the same category), or turkers marking the wrong odd-man-out, thus invalidating an otherwise correct puzzle.

Overall, the cost of the annotation and validation was below $500, for a large high-quality annotated resource and a smaller gold standard test corpus. Both corpora are made available.[6]

4 Taxonomy-Based Solvers

In this section, we show how to create odd-man-out solvers for taxonomies like WordNet (Miller, 1992).

Define a *taxonomy* as a triple (V, E, L), where (V, E) is a directed acyclic graph, and L maps each vertex V to a string. A simple example is shown in Figure 1, where the each vertex v is labeled with $L(v)$.

We create an odd-man-out solver from a taxonomy as follows:

- The *specificity* of a vertex v is defined as the reciprocal of the number of its descendants. For instance, the specificity of any leaf in Figure 1 is 1, while the specificity of the vertex labeled "element" is $\frac{1}{6}$.
- Given an odd-man-out puzzle $w_1, ..., w_n$, the *explanation* of word w_k is the vertex v of highest specificity such that: (i) for each word w_j such that $j \neq k$, there exists some descendent v' of v where $L(v') = w_j$, (ii) there does not exist a descendent v' of v such that $L(v') = w_k$. For instance, the explanation of **helium** with respect to the puzzle **helium, mercury, lead, silver, gold** is the node labeled "metallic element."
- If some word does not have an explanation, or if there is no word whose explanation is uniquely most specific, then the solver abstains from answering. Otherwise, the solver returns the word with the most specific explanation.

Using WordNet 3.0 (Miller, 1992; Fellbaum, 1998) as the taxonomy, the solver correctly solves

Category	Puzzle
construction	**crane**, *pelican*, excavator, hoist, upraise
guitar part	**fret**, *crying*, inlays, truss rod, neck
alcoholic drinks	**gin**, *poker*, vodka, tequila, wine
card games	**gin**, *vodka*, bridge canasta, uno

Table 1: Selection of examples from the crowdsourced Odd-Man-Out dataset. The polysemous seed word appears in bold, while the correct answer is in italics. The last two examples demonstrate how a polysemous word (*gin*) can participate in two different puzzles.

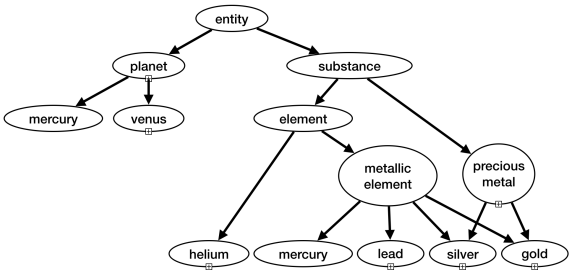

Figure 1: Example taxonomy.

[6] https://github.com/gabrielStanovsky/odd-man-out

Puzzle	Explanation
chicken, screwdriver, margarita, mimosa, daiquiri	mixed drink
silver, steel, brass, bronze, pewter	alloy
canoe, school, flock, herd, pack	animal group
nightgown, afternoon, morning, evening, midnight	abstraction
king, *president*, queen, prince, princess	leader
dinghy, *crab*, boat, canoe, raft	travel (verb)

Table 2: Some of the WordNet solver's answers and explanations on ANOMIACOMMONDEV. The solver's answer is in bold, while the correct answer is in italics.

40.6% of the ANOMIACOMMON puzzles, answering incorrectly for 13.4% and abstaining for 46.0% of the puzzles. Unsurprisingly (since WordNet focuses on common words), the WordNet solver gets only 1 out of 202 ANOMIAPROPER puzzles, abstaining from all the rest. On CROWDSOURCED843, the solver correctly solves 22.0% of the puzzles, answering 15.1% incorrectly and abstaining from the rest.

4.1 Error Analysis

A nice property of the taxonomy-based solver is that it provides an explanation for its answer (i.e. the "explanation" vertex defined above). In Table 2, we show some of the WordNet solver's answers and explanations (specifically we show the name of the WordNet synset corresponding to the explanation vertex). For 85.4% of its correct answers, the solver also returns the correct explanation (like the "mixed drink," "alloy," and "animal group" explanations). In the handful of cases when it does not, it finds either an incorrect or an overly vague explanation (like the "abstraction" explanation) that still results in the correct answer.

Typically, the incorrect answers result from incompleteness in the WordNet taxonomy. For instance, the word "king" is not a hyponym of "leader," even though "president," "queen," "prince," and "princess" all are. In the bottommost puzzle of Table 2, the error results from "raft" not being considered a watercraft (or a conveyance of any kind) by WordNet. Because of this, the solver finds a more tenuous connection between "crab" and three of the other words, leveraging a rare sense of "crab" as a verb meaning "to move like a crab."

5 Embedding-Based Solvers

In this section, we create odd-man-out solvers from collections of word embeddings. Specifically, we experiment with two types of embeddings: (1) traditional word embeddings, which map words to a single vector representation (Subsection 5.2), and (2) sense embeddings, which map words to a *set* of vectors, each pertaining to a different sense of the word (Subsection 5.3).

5.1 Embedding Evaluation Framework

For the sake of comparison, we use a common framework to create odd-man-out solvers based on traditional word embeddings and sense embeddings. In this framework, given n puzzle options, we find the subset of $n-1$ words of maximal similarity (given some similarity score), and return the excluded word as the odd-man-out.

Specifically, define an *embedding* as a function that maps every word to a *set* (possibly empty) of real vectors. Note that this definition is general enough to allow for sense embeddings as well as traditional word embeddings, for which the returned sets are either singletons or the empty set (in case the word is not in the embedding's vocabulary).

Next, given a similarity score σ that maps any vector pair to a real number and an embedding E, define the *cohesion* $\kappa_{\sigma,E}$ of a set of words W as:

$$\kappa_{\sigma,E}(W) = \max_{v_1 \in E(w_1),\ldots,v_n \in E(w_n)} \sum_{1 \le i < j \le n} \sigma(v_i, v_j) \quad (1)$$

Cohesion is undefined if any word $w \in W$ maps to the empty set. Throughout this paper we employ the widely used *cosine similarity* as our similiarity score.

Finally, given an embedding E and puzzle $W = \langle w_1, \ldots, w_n \rangle$, we create an odd-man-out solver as follows:

1. If $E(w_i)$ is the empty set for any puzzle choice $w_i \in W$, then the solver abstains from answering.

2. Otherwise, the solver returns the word $\hat{w} \in W$ whose omission from the puzzle word set

Embedding Map	Training Tokens	AnomiaCommon			AnomiaProper			Crowdsourced		
		C	W	A	C	W	A	C	W	A
ELMo clusters ($K = 5$)	1B + 2B*	**76.7**	**13.9**	**9.4**	**42.6**	**17.8**	**39.6**	**55.5**	**18.8**	**25.6**
w2v.googlenews	100B	61.9	25.2	12.9	40.1	14.9	45.0	46.3	28.8	24.9
glove.commoncrawl2	840B	60.9	23.8	15.4	32.2	14.4	53.5	47.1	28.4	24.6
glove.commoncrawl1	42B	57.4	29.2	13.4	30.7	17.8	51.5	40.1	36.3	23.7
glove.wikipedia	6B	54.5	24.3	21.3	29.2	10.9	59.9	42.7	29.0	28.4
Neelakantan	1B	35.2	25.7	39.1	18.3	14.4	67.3	32.6	27.3	40.2
w2v.freebase	100B	22.3	28.7	49.0	34.2	14.9	51.0	9.9	11.3	78.3
WordNet	-	40.6	13.4	46.0	0.5	0.0	99.5	22.0	15.1	63.0

Table 3: Performance (%Correct, %Wrong, %Abstained) of the different odd-man-out solvers on the ANOMIATEST and crowdsourcing datasets, using different vector directories. * We used ELMo clusters pretrained on 1B tokens and clustered on a 2B token Wikipedia dump.

yields maximal cohesion:

$$\hat{w} = \underset{w_i \in W}{\operatorname{argmax}} \ \kappa_\sigma(W \setminus \{w_i\}) \qquad (2)$$

5.2 Word Embeddings Solvers

Table 3 shows the results of embedding-based solvers on the Anomia and crowdsourced datasets, using several different pre-trained embedding maps. We find the best performance on ANOMIACOMMON and ANOMIAPROPER using the word2vec vectors trained on 100 billion tokens from the Google News corpus.[7]

An analysis of the best embedding-based solver (w2v.googlenews) on ANOMIACOMMONDEV shows that a high proportion of incorrect answers were polysemous (20 out of the 27 incorrect answers). A selection of these are shown in Table 4. We observe that for the "cocktails" puzzle, the word2vec solver zeroes in on "screwdriver" (whose dominant sense is the tool, not the cocktail), and for the "types of lettuce" puzzle, the solver selects "iceberg" (whose dominant sense is "large floating block of ice," not the lettuce).

To provide additional evidence of this bias, we came up with 5 cocktails whose dominant sense is the cocktail itself (mint julep, mai tai, mojito, martini, and bloody mary) and 5 polysemous cocktails (old fashioned, hurricane, cosmopolitan, zombie, and Manhattan). We then replaced "screwdriver" in the "cocktails" puzzle with each of these options and solved the resulting puzzle with the w2v.googlenews solver. In all 5 monosemous instances, the correct answer of "chicken" was selected. In 4 of the 5 polysemous instances (the exception being "Manhattan"), the polysemous replacement was selected. We repeated this experiment with the "groups of animals" puzzle, and again all 5 monosemous replacements yielded the correct answer, while 3 of 5 polysemous replacements were incorrectly chosen by the solver.

In a more rigorous experiment, we randomly generated 500 puzzles in the following way. Given a category (hyponym), we came up with 10 instances (hypernyms) of that category. For example, the category "type of transport" yielded: train, car, bus, airplane, helicopter, boat, ferry, taxi, tram, and monorail. We did this for 10 categories, yielding a total of 100 words. From these 10 lists, we randomly generated 500 puzzles by sampling 4 words from one list and 1 word from another. We call this puzzle set HYPERNYMS500.

The w2v.googlenews solver performs impressively on HYPERNYMS500, getting 90.8% correct, with only 46 incorrect answers (and no abstentions). Tellingly however, only 3 words are responsible for 67% of the incorrect answers: saw (as a kind of tool), lead (as a kind of metal), and rose (as a kind of flower). All three of these words have at least one dominant alternate sense. Given that there are 100 possible incorrect answers (which all appear with roughly equal frequency in the puzzles), the fact that only 3 of them comprise 67% of the incorrect answers suggests that the inability of vector directories to model multiple senses is an Achilles heel.

5.3 ELMo Sense Vectors

The previous analysis suggests that embeddings that explicitly model multiple senses may be important for the odd-man-out task. In this section,

[7] https://code.google.com/archive/p/word2vec

Category	Puzzle
cocktails	**screwdriver**, *chicken*, margarita mimosa, daiquiri
mammals	**bear**, *cobra*, cat dog, leopard
nocturnal animals	**bat**, *robin*, owl raccoon, coyote
military ranks	**private**, *judo*, general corporal, colonel
whales	**blue**, *grizzly*, humpback orca, beluga
groups of animals	**school**, *canoe*, flock herd, pack
shades of blue	**navy**, *amber*, cobalt azure, sky
aquatic animals	**seal**, *horse*, whale manatee, dolphin
bird sounds	**tweet**, *snore*, chirp cluck, quack
types of lettuce	**iceberg**, *serrano*, romaine butterhead, bibb
political parties	**green**, *garden*, democratic republican, libertarian

Table 4: Selection of errors made by the w2v.googlenews solver on ANOMIACOMMONDEV, indicating a tendency to choose polysemous words as the odd-man-out. The incorrect selection is in bold, while the correct answer is in italics.

we investigate this further.

Background: sense vectors There is a significant literature on how to learn a one-to-many mapping from words to vector representations. An early paradigm (Reisinger and Mooney, 2010; Huang et al., 2012; Liu et al., 2015; Wu and Giles, 2015) took a 2-pass approach. First, they clustered contexts of a target word like "bank." For instance, "the bank had an ATM" and "I got money from the bank" might fall into one cluster, while "I fished at the river bank" might fall into a second cluster. Second, they annotated each instance of the target word with its sense cluster and then learned a standard one-to-one vector mapping from the annotated corpus. For instance, they would learn vector representations using the sentences "the bank-1 had an ATM," "I got money from the bank-1," and "I fished at the river bank-2." Other researchers (Neelakantan et al., 2014; Tian et al., 2014; Chen et al., 2014; Li and Jurafsky, 2015; Bartunov et al., 2016) focused on modifying the vector learning model itself, typically the skip-gram model (Mikolov et al., 2013b), to directly learn multiple embeddings for each word. Additional work focused on using the technique of retrofitting (Faruqui et al., 2014) to adapt pretrained word vectors into sense vectors using auxiliary resources like WordNet (Jauhar et al., 2015) or parallel corpora (Ettinger et al., 2016). Other work (Guo et al., 2014; Suster et al., 2016; Upadhyay et al., 2017) used parallel corpora as the main signal for learning sense vectors.

Background: ELMo Recently, Peters et al. (2018) introduced the concept of Embeddings from Language Models (ELMo). ELMo dynamically represents each word based on the context with which it appears, achieved by representing a word in a sentence using its representation from a pretrained bidirectional Language Model (biLM), encoded using a bi-directional RNN (Schuster and Paliwal, 1997). Subsequently, the same word may get different representations in different contexts. For example, the representation of "*bank*" may differ between "*She fished by the river **bank***" and "*She deposited her check at the nearest **bank***". While ELMo embeddings were recently proven extremely beneficial in various sentence-level tasks (e.g., semantic role labeling, question answering, and textual entailment), many lexical tasks require the interpretation of words out of context, to which ELMo cannot be readily applied. We suggest to port ELMo's effectiveness back to the context-free setting, and propose a first approach for doing so.

Unsupervised ELMo clustering We compute pre-trained ELMo embedding (using the AllenNLP framework (Gardner et al., 2018)) *in context* of sentences in a large corpus C, while recording the observed representation $r_{w,s} \in \mathbb{R}^d$ of each word w, along with the corresponding context in which it appeared, i.e., the sentence $s \in C$. The result of this process is an embedding space *per word*, $r_w = \{r_{w,s} \mid s \in C, w \in s\}$. Ideally, similar senses of w would appear in similar contexts, and would therefore be closer in r_w, while different senses would be further apart. Following this intuition, we cluster each word to K clusters, using the k-means algorithm (Lloyd, 1982), whereby every vector in a cluster is interpreted as pertaining to the same sense of w. We collect the centroid

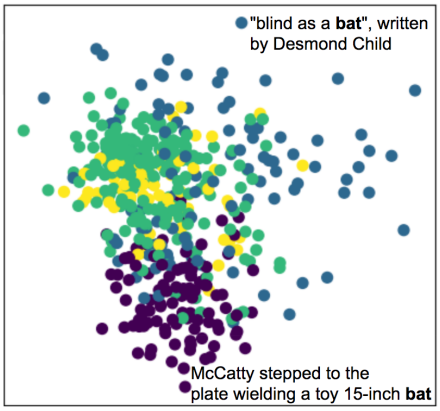

Figure 2: An example of a 2d projection of an ELMo embedding space for the word "bat" with 4 senses (denoted by different colors). Two example sentence excerpts appear next to their respective cluster.

vector w_i (where $1 \leq i \leq K$) of each cluster as a "sense vector" of w. Using $K = 1$, we get a single representation per word, averaging the different senses of w, comparable to "traditional" word embeddings. Conversely, we can get senses of arbitrarily fine granularity by increasing the number of clusters. These sense representations can then be readily plugged in to the embedding solver, as described in Subsection 5.1. We computed the sense vectors using a 2 billion token (97 Million sentences) Wikipedia dump from January 2018, which was extracted using WikiExtractor,[8] and tokenized with the SpaCy 2.0 toolkit (Honnibal and Montani, 2017).[9] These pre-computed vectors, which are readily applicable for other tasks, are made publicly available. Figure 2 depicts the resulting embedding space and clustering for the word "bat".

Evaluation We start by estimating an ideal value for K (the number of clusters). In Figure 3 we evaluate the performance of the ELMo clustering method on the expert development set, using different values for K, compared to the second best performing vectors on that dataset (Word2Vec). Several observations can be made based on this analysis. First, using $K = 1$, Word2Vec outperforms ELMo. This may be explained due to Word2Vec's larger training set (100B tokens versus only 1B). However, as K increases, ELMo clusters outperform the baseline,

[8] https://github.com/attardi/wikiextractor

[9] https://spacy.io

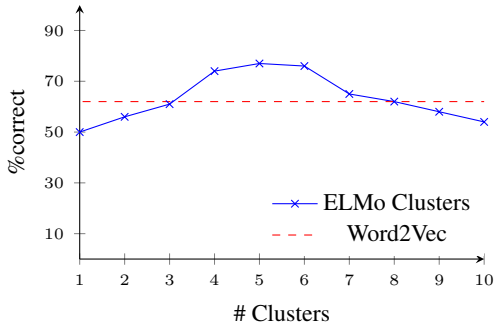

Figure 3: Performance of the ELMo cluster model (blue line) by the varying number of cluster (K), compared to the best performing baseline (word2vec), in the red horizontal line.

reaching its maximum performance at $K = 5$, indicating that a finer level of sense granularity is beneficial for focusing on the intended word sense. Inversely, having too many clusters hurts performance. This may be due to over-specification, harming the sense generalization obtained by a smaller number of clusters. Based on this tuning, we fixed the value of K to 5, and repeated the experiments from Section 5 (see the first row in Table 3). ELMo sense vectors clearly outperform all previous baselines on all Odd-Man-Out datasets. This improved performance can be attributed both to ELMo's better ability to capture context, as well as to the finer sense representation, as opposed to the single representation per word in most of the other baselines. We also evaluated another publicly available collection of sense embeddings (Neelakantan et al., 2014), but it did not perform on par with solvers based on conventional single-sense embeddings.

6 Discussion: Hypernyms vs. Other Associations

The attentive reader may have wondered why the w2v.googlenews solver performed so much better on the HYPERNYMS dataset (over 90% correct) than on the ANOMIACOMMON dataset (approximately 62% correct). Hypothesizing that embedding-based solvers can identify hypernym-hyponym relationships more easily than other associations, we created another odd-man-out dataset using the same methodology as HYPERNYMS.

This time, given a *color*, we came up with 8 to 13 objects that are typically associated with that color. For example, the category "yellow" yielded

(among others): taxi, canary, corn kernel, daffodil, lemon, sun, and school bus. We did this for 10 colors. From these 10 lists, we again randomly generated 500 puzzles by sampling 4 words from one list and 1 word from another. We call this puzzle set COLORS500.

The w2v.googlenews solver performed considerably worse on COLORS500 than on HYPERNYMS500. Out of the puzzles it attempted, it guessed only 30% correctly (compared to the random chance baseline of 20%). There are several possible reasons why this dataset may be more difficult for an embedding-based solver. Possibly colors are not easily identifiable using embeddings built from language cues (maybe people do not often explicitly talk about how lemons are yellows). But perhaps it is also true that using the cosine similarity of the embeddings is not a good way to spot non-fundamental properties (like color) that link a group of words. We leave open how best to identify more oblique relationships.

7 Conclusion

We presented a new task for the evaluation of lexical resources, the Odd-Man-Out task. We showed that the task can be annotated reliably on a large scale. Following the creation of several Odd-Man-Out datasets, we conducted analyses showing that current word representations are suboptimal, especially in the presence of polysemous words. We concluded with a novel ELMo clustering sense-embedding technique which surpasses all baselines on the Odd-Man-Out task.

Acknowledgements

We would like to thank Santosh Divvala, Fereshteh Sadeghi, Marco Valenzuela, and Isaac Cowhey, who initiated this project during an AI2 hackathon.

References

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. Linear algebraic structure of word senses, with applications to polysemy. *CoRR*, abs/1601.03764.

Sergey Bartunov, Dmitry Kondrashkin, Anton Osokin, and Dmitry P. Vetrov. 2016. Breaking sticks and ambiguities with adaptive skip-gram. In *AISTATS*.

Christian Biemann. 2013. Creating a system for lexical substitutions from scratch using crowdsourcing. *Language Resources and Evaluation*, 47:97–122.

Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. 2014. A unified model for word sense representation and disambiguation. In *EMNLP*.

Allyson Ettinger, Philip Resnik, and Marine Carpuat. 2016. Retrofitting sense-specific word vectors using parallel text. In *HLT-NAACL*.

Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166*.

Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. 2016. Problems with evaluation of word embeddings using word similarity tasks. In *RepEval@ACL*.

Christiane D. Fellbaum. 1998. Book reviews: Wordnet: An electronic lexical database.

Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. 2001. Placing search in context: the concept revisited. *ACM Trans. Inf. Syst.*, 20:116–131.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Anna Gladkova and Aleksandr Drozd. 2016. Intrinsic evaluations of word embeddings: What can we do better? In *RepEval@ACL*.

Yoav Goldberg. 2015. A primer on neural network models for natural language processing. *arXiv preprint arXiv:1510.00726*.

Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. 2014. Learning sense-specific word embeddings by exploiting bilingual resources. In *COLING*.

Matthew Honnibal and Ines Montani. 2017. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*.

Eric H. Huang, Richard Socher, Christopher D. Manning, and Andrew Y. Ng. 2012. Improving word representations via global context and multiple word prototypes. In *ACL*.

Sujay Kumar Jauhar, Chris Dyer, and Eduard H. Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *HLT-NAACL*.

David Jurgens, Saif Mohammad, Peter D. Turney, and Keith J. Holyoak. 2012. Semeval-2012 task 2: Measuring degrees of relational similarity. In *SemEval@NAACL-HLT*.

Gerhard Kremer, Katrin Erk, Sebastian Padó, and Stefan Thater. 2014. What substitutes tell us - analysis of an "all-words" lexical substitution corpus. In *EACL*.

Jiwei Li and Daniel Jurafsky. 2015. Do multi-sense embeddings improve natural language understanding? In *EMNLP*.

Yang Liu, Zhiyuan Liu, Tat-Seng Chua, and Maosong Sun. 2015. Topical word embeddings. In *AAAI*.

Stuart P. Lloyd. 1982. Least squares quantization in pcm. *IEEE Trans. Information Theory*, 28:129–136.

Suresh Manandhar, Ioannis P. Klapaftis, Dmitriy Dligach, and Sameer Pradhan. 2010. Semeval-2010 task 14: Word sense induction and disambiguation. In *SemEval@ACL*.

Diana McCarthy and Roberto Navigli. 2007. Semeval-2007 task 10: English lexical substitution task. In *SemEval@ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*.

George A. Miller. 1992. Wordnet: A lexical database for english. *Commun. ACM*, 38:39–41.

Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41:10:1–10:69.

Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew D McCallum. 2014. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *CoRR*, abs/1802.05365.

Joseph Reisinger and Raymond J. Mooney. 2010. Multi-prototype vector-space models of word meaning. In *HLT-NAACL*.

Herbert Rubenstein and John B. Goodenough. 1965. Contextual correlates of synonymy. *Commun. ACM*, 8:627–633.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing*, 45:2673–2681.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. 2016. Improving hypernymy detection with an integrated path-based and distributional method. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2389–2398, Berlin, Germany. Association for Computational Linguistics.

Yifan Sun, Nikhil Rao, and Weicong Ding. 2017a. A simple approach to learn polysemous word embeddings. *CoRR*, abs/1707.01793.

Yifan Sun, Nikhil Rao, and Weicong Ding. 2017b. A simple approach to learn polysemous word embeddings. *CoRR*, abs/1707.01793.

Simon Suster, Ivan Titov, and Gertjan van Noord. 2016. Bilingual learning of multi-sense embeddings with discrete autoencoders. In *HLT-NAACL*.

Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. 2014. A probabilistic model for learning multi-prototype word embeddings. In *COLING*.

Shyam Upadhyay, Kai-Wei Chang, Matt Taddy, Adam Tauman Kalai, and James Y. Zou. 2017. Beyond bilingual: Multi-sense word embeddings using multilingual context. In *Rep4NLP@ACL*.

Zhaohui Wu and C. Lee Giles. 2015. Sense-aware semantic analysis: A multi-prototype word representation model using wikipedia.